CSE 4251: Lab 4

Due Date: Nov 15

Objectives

- Get familiar with the system working environment and basic commands
- Get familiar with shell programming and script writing
- Incorporate other Unix/Linux commands and script constructs

Instructions

Exercise 1 Description

For this exercise you are required to make a copy of the following script into a file named ***guess.sh***:

```
#!/bin/tcsh -f

echo "guess the number"
set answer = $<

set goal= `tr -dc A-Za-z0-9 < /dev/urandom | od -d | head -c 10 | tr -d 0 |
tr -d ' '`

set count = 1
while ($answer != $goal )
    if ($answer < $goal ) then
        echo "too small"
    else
        echo "too large"
    endif
    @ count ++
    set answer = $<
end

echo "correct!"
echo "using $count rounds"
```

a. Study the script and be sure to understand it. Review the slides about Shell Overview and Shell Programming with the description of what it does. Your task is to **incorporate comments in different parts of the script to explain what it does.**

b. Execute the script and **correct any error it may have to run it**. You will have to change permissions to the shell script file to make it executable (remember the *chmod* command).
c. You are required to modify the script, make a copy with the name ***guess_game.sh*** and implement the following changes:
    1. Create a main menu, so when you execute the script the user will have two options (try the command clear to clean the screen):

```
1.Run the guessing game
2.Exit
```

    2. When the user selects run the game, the game will do the same thing with an addition. It should now keep a record of each number of rounds.
    3. When finishing the game, the script returns to the main menu, so the user can play again
    4. When the user selects Exit, the script before ending will print out to the screen the average rounds for all games that were played:

```
Exiting the guessing game
Your average of rounds is: calculated average
```

d. See if using the *exit* status of a command may be of any help.

The script must have a comment section with the script name(s), your name, the date, and a short description of the script.

When you have it working correctly, use the command *script* to save the evidence or executing correctly and submit it. Name that file ***Lab4_1.txt.***

Exercise 2 Description

For this exercise you are required to make a copy of the following script into a file ***dirtree.csh***, that prints a directory tree:

```
#!/bin/tcsh -f
if($#argv == 0) then
        set thisdir="."
else
        set thisdir=$argv[1]
endif
if($?TREEPREFIX) then
        set prefix="$TREEPREFIX"
else
        set prefix=""
endif
echo "$prefix|"
set filelist=`ls -A $thisdir`
foreach file ($filelist)
        echo "${prefix}|------${file}"
```

```
        if(-d "$thisdir/$file") then
            if($file == $filelist[$#filelist]) then
                        setenv TREEPREFIX "${prefix}     "
                else
                        setenv TREEPREFIX "${prefix}|    "
            endif
            echo $0 "$thisdir/$file"
        endif
end
echo "$prefix"
```

a. Study the script and be sure to understand it. Review the slides about Shell Overview and Shell Programming with the description of what it does. **Incorporate comments in different parts of the script to explain what it does.**

b. Execute the script and correct any error it may have to run it. You will have to change permissions to the shell script file to make it executable (remember the *chmod* command).

c. You are required to modify the script, make a copy with the name ***dirtree_explorer.sh*** and implement the following changes:

1. Create a main menu, so when you execute the script the user will have these options:

```
The current directory is: /usr/xxx/xxxxxx/
1.Enter a directory name
2.Print the directory tree on terminal
3.Print the directory tree to a file
4.Exit
```

2. When the user runs the script, it will show the home directory at the top of the main menu, unless a directory was passed as an argument to the script, in that case it will show that one as a current directory in the menu
3. When the user selects *Enter a directory name*, the script will ask user to input a directory name and capture that into a variable and then it will show that one as a current directory.
4. When the user selects *Print the directory tree on terminal*, the script will print the file system structure under this directory in the form of a tree (as the original script did) and then returns to the main menu.
5. When the user selects *Print the directory tree to a file*, the script will ask the user for the name of the file and then will save the file system structure under this directory in the form of a tree (as the original script did) to that file and then returns to the main menu. You may place the file in the directory where the shell script is. Remember that you may send data to a file and appended with redirection operators (> or >>).
6. If the option 4 is selected, the script terminates.

The script must have a comment section with the script name, your name(s), the date, and a short description of the script.

When you have it working correctly, use the command *script* to save the evidence and submit it. Name that file ***Lab4_2.txt***

Submission Instructions

To submit, create a single zip file named with the format of **Lab4.zip** that contains the files from the different exercises. Please note this zip file should have only **4 files:**
guess_game.sh
dirtree_explorer.sh
Lab4_1.txt
Lab4_2.txt
Points will be deducted for extra files, incorrect naming of the zip file, and non-readable script recorded files.